C# 프로그래밍

- 9 주차 (2강)

SEJONG UNIVERSITY

- 인덱서
- foreach 문이 가능한 객체
- 인터페이스

인덱서 (Indexer)

- 인덱스를 이용해서 객체 내의 데이터에 접근하게 해주는 프로퍼티
  - ✓ 객체를 마치 배열처럼 사용할 수 있게 지원
  - ✓ 프로퍼티처럼 식별자를 따로 가지지 않음
  - ✓ 인덱스를 통해 객체 내의 데이터에 접근

인덱서 선언 형식:

```
class 클래스이름
{
    한정자 인덱서형식 this[형식 index]
    {
        get
        {
            // index 를 이용하여 내부 데이터 반환
        }

        set
        {
            // index 를 이용하여 내부 데이터 저장
        }
    }
}
```

인덱서 사용 예:

```
private int[] array;
public MyList() { array = new int[3]; }

public int this[int index]  // 인덱서
{
    get {
        return array[index]; }

    set {
        if (index >= array.Length) {
            Array.Resize<int>(ref array, index + 1);
            Console.WriteLine("Array Resized : {0}", array.Length);
        }
        array[index] = value;
    }
}
```

인덱서 예제 코드

```
class MyList
{
    private int[] array;
    public MyList() { array = new int[3]; }

    public int this[int index]  // 인덱서
    {
        get {
            return array[index]; }

        set {
            if (index >= array.Length)
            {
                Array.Resize<int>(ref array, index + 1);
                Console.WriteLine("Array Resized : {0}", array.Length);
            }
            array[index] = value;
        }
    }

    public int Length { get { return array.Length; } }
}
```

```
static void Main(string[] args)
{
    MyList list = new MyList();
    for (int i = 0; i < 5; i++)
        list[i] = i;    // 인덱스를 통해 데이터 입력

    for (int i = 0; i < list.Length; i++)
        Console.WriteLine(list[i]);
        // 인덱스를 통해 데이터 접근
}
```

출력 결과:

```
Array Resized : 4
Array Resized : 5
0
1
2
3
4
```

- 인덱서
- foreach 문이 가능한 객체
- 인터페이스

foreach 가 가능한 객체

- foreach 문의 특징
  - ✓ for 문과 달리 요소의 위치를 위한 인덱스 변수를 선언할 필요가 없음
  - ✓ 세미콜론, 조건문, 증감식을 사용할 필요 없음

for 문 사용 예제 :

```
int[] arr = new int[] { 11, 22, 33, 44, 55, 66, 77 };

for (int i = 0; i < arr.Length; i++)
    Console.WriteLine(arr[i]);
```

foreach 문 :

```
int[] arr = new int[] { 11, 22, 33, 44, 55, 66, 77 };

foreach (int e in arr)
    Console.WriteLine(e);
```

출력 결과:

```
11
22
33
44
55
66
77
```

foreach 가 가능한 객체

- foreach 문은 IEnumerable 을 상속하는 형식만 지원
  - ✓ IEnumerable 인터페이스는 GetEnumerator () 메소드를 가짐

메소드	설명
IEnumerator GetEnumerator( )	IEnumerator 형식의 객체를 반환

- yield 문의 이용
  - ✓ 컴파일 시 자동으로 IEnumerator 를 상속하는 클래스를 생성
  - ✓ yield return 문
    - 현재 메소드(GetEnumerator ())의 실행을 일시 정지하고 호출자에게 결과를 반환
    - 메소드가 다시 호출되면, 일시 정지된 실행을 복구
    - yield return 또는 yield break 문을 만날 때까지 나머지 작업을 실행

yield 키워드를 이용하는 간단한 예제

```
class MyEnumerator
{
    int[] numbers = { 1, 2, 3, 4 };
    public IEnumerator GetEnumerator()
    {
        yield return numbers[0];
        yield return numbers[1];
        yield return numbers[2];
        yield break;                // GetEnumerator() 종료
        yield return numbers[3]; // 실행되지 않음
    }
}
```

```
static void Main(string[] args)
{
    var obj = new MyEnumerator();
    foreach (int i in obj)
        Console.WriteLine(i);
}
```

출력 결과:
1
2
3

foreach 가 가능한 객체

- GetEnumerator ( ) 메소드는 IEnumerator 형식의 객체를 반환
  - ✓ IEnumerator 인터페이스의 메소드 및 프로퍼티 목록

메소드	설명
boolean MoveNext ( )	다음 요소로 이동. 컬렉션의 끝을 지난 경우에는 false, 이동이 성공한 경우는 true 반환.
void Reset ( )	컬렉션의 첫 번째 위치의 '앞'으로 이동. 첫 번째 위치가 0번인 경우 Reset( )을 호출하면 -1번으로 이동. 첫 번째 위치로의 이동은 MoveNext( )를 호출한 다음에 이루어짐
Object Current {get;}	컬렉션의 현재 요소를 반환

  - ✓ MoveNext ( ), Reset ( ) 메소드와 Current 프로퍼티를 구현하면 해당 클래스의 객체는 IEnumerator 형으로 변환 가능
    - 해당 클래스에서 IEnumerable이 요구하는 GetEnumerator ( ) 메소드를 구현할 때는 자기 자신(this)을 반환하면 됨

foreach 가 가능한 객체관련 예제 (1)

```
class MyList : IEnumerator, IEnumerable
{
    private int[] array;
    int position = -1;
    public MyList() { array = new int[3]; }

    public int this[int index]  // 인덱서
    {
        get { return array[index]; }
        set {
            if (index >= array.Length)
            {
                Array.Resize<int>(ref array, index + 1);
                Console.WriteLine("Array Resized : {0}", array.Length);
            }
            array[index] = value; }
    }

    // IEnumerator 멤버
    public object Current {
        get { return array[position]; }
    }
    // IEnumerator 멤버
    public bool MoveNext()
    {
        if(position == array.Length-1)
        {
            Reset();
            return false;
        }
        position++;
        return (position < array.Length);
    }

    // IEnumerator 멤버
    public void Reset()
    {
        position = -1;
    }

    // IEnumerable 멤버
    public IEnumerator GetEnumerator()
    {
        return this;
    }
}
```

foreach 가 가능한 객체관련 예제 (2)

```
static void Main(string[] args)
{
    MyList list = new MyList();
    for (int i = 0; i < 5; i++)
        list[i] = i;          // 인덱스를 통해 데이터 입력

    foreach (int e in list) // for문 대신에 foreach문 사용
        Console.WriteLine(e);
}
```

출력 결과:

```
Array Resized : 4
Array Resized : 5
0
1
2
3
4
```

- 인덱서
- foreach 문이 가능한 객체
- 인터페이스

인터페이스(Interface) 선언

- 인터페이스는 interface 키워드를 이용해서 선언
- 메소드, 이벤트, 인덱서, 프로퍼티만을 가질 수 있음
  - ✓ 메소드, 이벤트, 인덱서, 프로퍼티의 구현부는 없음
  - ✓ 접근 제한 한정자(e.g. private) 사용 불가
- 인터페이스는 인스턴스를 만들 수 없음
  - ✓ 인터페이스를 상속받는 클래스의 인스턴스를 만드는 것은 가능

인터페이스의 선언

```
interface 인터페이스이름
{
    반환_형식 메소드이름1(매개변수_목록);
    반환_형식 메소드이름2(매개변수_목록);
    반환_형식 메소드이름3(매개변수_목록);
    // ...
}
```

인터페이스 사용 예:

```
interface ILogger
{
    void WriteLog(string message);
}
```

인터페이스 사용

- 인터페이스를 상속받는 파생 클래스
  - ✓ 인터페이스에 선언된 모든 메소드(및 프로퍼티)를 구현 해야함
  - ✓ 해당 메소드들은 public 한정자로 수식
- 인터페이스는 인스턴스를 만들 수 없지만 참조는 만들 수 있음
  - ✓ 참조에 파생 클래스 객체의 위치를 저장

ILogger 인터페이스를 상속받는 파생 클래스

```
interface ILogger { void WriteLog (string message); }
class ConsoleLogger : ILogger
{
    public void WriteLog(string msg)
    {
        Console.WriteLine("{0} {1}", DateTime.Now.ToLocalTime(), msg);
    }
}
```

인터페이스 참조에 파생 클래스 객체의 위치 저장

```
static void Main(string[] args)
{
    ILogger logger = new ConsoleLogger();
    logger.WriteLog("Hello, World");
}
```

출력 결과:

```
2021-04-29 오전 6:15:33 Hello, World
```

인터페이스는 약속 이다

- USB 포트를 예를 들어 설명하면, PC와 주변기기는 USB 포트라는 약속을 따름
  - ✓ USB 플래시 메모리를 꽂아 넣으면 저장 장치로 사용
  - ✓ 키보드나 마우스를 꽂으면 입력 장치로 사용
- 인터페이스도 소프트웨어 내에서 USB와 같은 역할 수행
  - ✓ 클래스가 따라야 하는 약속
  - ✓ 인터페이스로부터 파생될 클래스가 어떤 메소드를 구현해야 할지를 정의
- ILogger 인터페이스는 파생클래스에 WriteLog() 메소드 구현을 강제
  - ✓ ConsoleLogger 파생클래스를 선언하여 콘솔에 로그 출력하도록 WriteLog() 구현가능
  - ✓ 또한, ILogger를 상속받는 새로운 클래스를 선언하여 파일에 로그출력도 가능

```
interface ILogger { void WriteLog (string message); }
class ConsoleLogger : ILogger
{
    public void WriteLog(string msg)
    {
        Console.WriteLine("{0} {1}", DateTime.Now.ToLocalTime(), msg);
    }
}
```

ILogger 인터페이스 사용예제

- ClimateMonitor 클래스
  - ✓ 사용자로부터 온도를 반복적으로 입력 받아 기록
  - ✓ 로그를 저장하는 방식은 개발자의 결정에 따라 달라질 수 있도록 구현

```
class ClimateMonitor
{
    private ILogger logger;
    // 생성자의 매개변수는 로그저장 방식을 결정
    public ClimateMonitor(ILogger logger) { this.logger = logger; }

    public void start() {
        while(true) {
            Console.Write("온도를 입력해주세요: ");
            string temperature = Console.ReadLine();
            if (temperature == "")
                break;

            logger.WriteLog("현재 온도: " + temperature);
        }
    }
}
```

ILogger 인터페이스 사용예제

- ILogger 인터페이스를 상속 받는 ConsoleLogger 클래스
  - ✓ WriteLog() 메소드 구현 필수
  - ✓ 콘솔에 로그를 출력

```
class ConsoleLogger : ILogger
{
    public void WriteLog(string message)
    {
        Console.WriteLine("{0} {1}", DateTime.Now.ToLocalTime(), message);
    }
}
```

ConsoleLogger 객체를 생성자에 인수로 넘김

```
ClimateMonitor monitor = new ClimateMonitor(new ConsoleLogger());
monitor.start();
```

출력 결과:

```
온도를 입력해주세요: 33
2021-04-29 오전 6:52:49 현재 온도: 33
온도를 입력해주세요: 28
2021-04-29 오전 6:52:52 현재 온도: 28
온도를 입력해주세요: 76
2021-04-29 오전 6:52:58 현재 온도: 76
온도를 입력해주세요:
```

ILogger 인터페이스 사용예제

- ILogger 인터페이스를 상속 받는 FileLogger 클래스
  - ✓ WriteLog() 메소드 구현 필수
  - ✓ 파일에 로그를 출력

```
class FileLogger : ILogger
{
    private StreamWriter writer;

    public FileLogger(string path)
    {
        writer = File.CreateText(path);
        writer.AutoFlush = true;
    }

    public void WriteLog(string message)
    {
        writer.WriteLine("{0} {1}", DateTime.Now.ToLocalTime(), message);
    }
}
```

입력(콘솔):
온도를 입력해주세요: 34
온도를 입력해주세요: 28
온도를 입력해주세요: 77
온도를 입력해주세요:

출력 결과 (MyLog.txt):
2021-04-29 오전 6:52:39 현재 온도: 34
2021-04-29 오전 6:52:43 현재 온도: 28
2021-04-29 오전 6:52:45 현재 온도: 77

FileLogger 객체를
생성자에 인수로 넘김

```
ClimateMonitor monitor = new ClimateMonitor(new FileLogger("MyLog.txt"));
monitor.start();
```

Thank you!